Phân Tích Dữ Liệu Thực Tế với Python

Bài 5.1: Numpy & Text & Datetime

Quang-Khai Tran, Ph.D
CyberLab, 03/2023

(Ảnh: Internet)

CYBERLEARN
ĐÀO TẠO CHUYÊN GIA LẬP TRÌNH

CYBERSOFT
ĐÀO TẠO CHUYÊN GIA LẬP TRÌNH

CYBERLAB
ĐÀO TẠO CHUYÊN GIA LẬP TRÌNH

Nội dung

1. Giới thiệu thư viện Numpy
2. Cơ bản về xử lý dữ liệu text
3. Cơ bản về dữ liệu thời gian
4. Thảo Luận & Bài tập

Review: Các loại dữ liệu cơ bản trong Python

Loại	Từ khóa	Ví dụ
Dạng text	`str`	`"cyberlab"`
Dạng số (numeric)	`int, float, complex`	`10, 10.1`

Dạng chuỗi (sequence)	`list, tuple, range`	`L = ["Nam", "Lan", "Peter"]`
		`T = (15, 20, 10)`
		`R = [0, 1, 2, 3, 4, 5]`

Dạng ánh xạ (mapping)	`dict`	`P1 = {"name": "Nam", "age": 23}`
Dạng tập hợp	`set, frozenset`	`{1, 2, 3, 4, 5}`
Dạng boolean	`bool (True, False)`	`ok = True`
Dạng nhị phân	`bytes, bytearray, memoryview`	

Phần 1. Giới thiệu thư viện Numpy

1.1. Cơ bản về Numpy
1.2. Một số hàm tính toán, sắp xếp
1.3. Một số khởi tạo đặc biệt
1.4. Tìm kiếm, so sánh và đếm

1 Giới thiệu thư viện Numpy

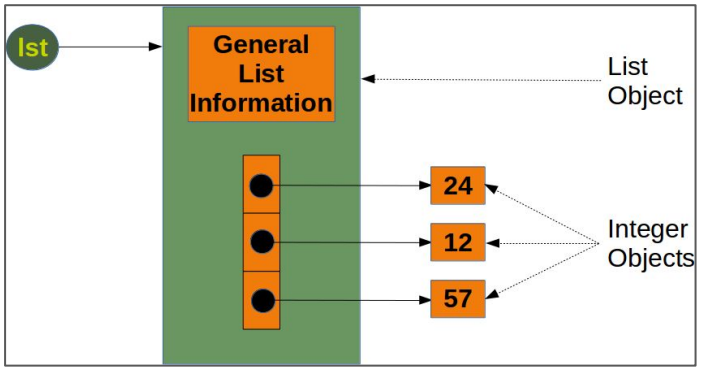
- Numpy có thể xem là công cụ cơ bản nhất (thư viện lõi của Python) và tốt nhất để xử lý dữ liệu dạng số và tính toán khoa học trong Python
- Numpy đồng thời là nền tảng của các công cụ nâng cao: Matplotlib, Pandas và nhiều công cụ tính toán khoa học khác
- Thành phần quan trọng nhất là mảng n-chiều (n-dimensional array)

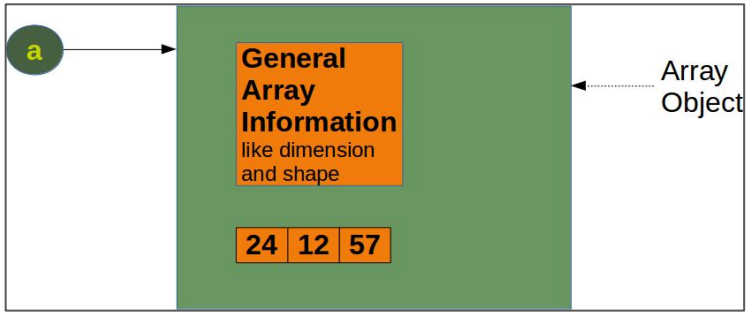
1 Giới thiệu thư viện Numpy

- Routines trong Numpy:
  là các thủ tục được phân chia theo loại thao tác hoặc tính năng
  - Tạo mảng, truy cập mảng
  - Chỉnh sửa mảng
  - Các thao tác toán học
  - Các thao tác tìm kiếm, sắp xếp, random
  - Các hàm thống kê
  - …
- Danh sách các routines của Numpy:
  https://numpy.org/doc/stable/reference/routines.html

1 Giới thiệu thư viện Numpy

- So sánh List vs. Array
- Xử lý dữ liệu dạng số với các hàm của Python có thể:
  - Rất chậm
  - Tiêu tốn nhiều bộ nhớ





Giá trị "khuyết": NaN (Not a Number)

- Dùng để biểu diễn thành phần chưa được định nghĩa giá trị trong mảng
- Khi mảng có giá trị NaN, các hàm tính toán sẽ cho kết quả NaN
- **Khi đó cần các hàm khác, hoặc cần loại bỏ/thay thế giá trị NaN**
  - Hàm khác: np.nansum(), np.nanmax()...
  - Bỏ dòng có giá trị NaN: np.dropna()
  - Thay giá trị NaN: np.fillna(), np.nan_to_num()

```
import numpy as np
arr = np.array([1, np.nan, 3, 6, np.nan])
np.isnan(arr)
# [False True False False True]
sum(~np.isnan(arr))
```

Các lớp và hàm cơ bản:

- Lớp numpy.ndarray()
- Hàm tạo mảng Numpy:
  - numpy.array()
  - numpy.asarray()

```
#  Tip: dùng hàm set_printoptions để giới hạn số chữ số
#       sau dấu chấm
import numpy as np
np.set_printoptions(precision=3)
```

Các loại dữ liệu (data type):

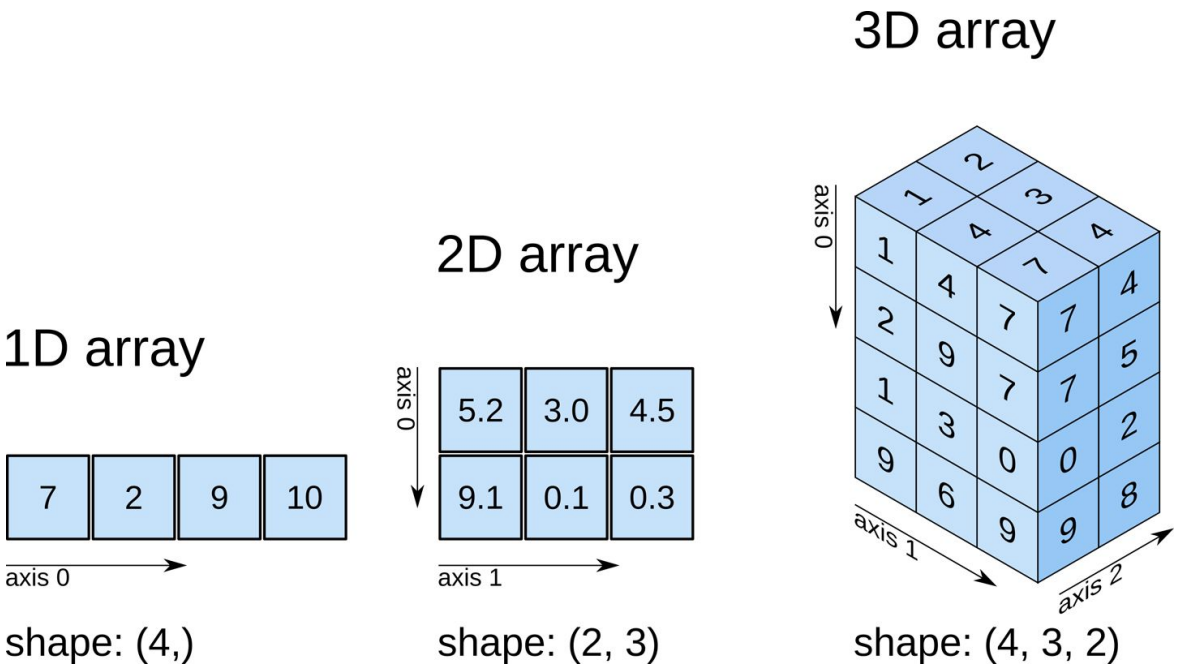
- Dạng số
- Dạng chuỗi (np.char)
- Dạng datetime

Basic Type	Available NumPy types	Comments
Boolean	bool	Elements are 1 byte in size
Integer	int8, int16, int32, int64, int128, int	int defaults to the size of int in C for the platform
Unsigned Integer	uint8, uint16, uint32, uint64, uint128, uint	uint defaults to the size of unsigned int in C for the platform
Float	float32, float64, float, longfloat,	Float is always a double precision floating point value (64 bits). longfloat represents large precision floats. Its size is platform dependent.
Complex	complex64, complex128, complex	The real and complex elements of a complex64 are each represented by a single precision (32 bit) value for a total size of 64 bits.
Strings	str, unicode	Unicode is always UTF32 (UCS4)
Object	object	Represent items in array as Python objects.
Records	void	Used for arbitrary data structures in record arrays.

Data Type	Description
bool_	Boolean (True or False) stored as a byte
int8	Byte (-128 to 127)
int16	Integer (-32768 to 32767)
int32	Integer (-2.15E-9 to 2.15E+9)
int64	Integer (-9.22E-18 to 9.22E+18)
uint8	Unsigned integer (0 to 255)
uint16	Unsigned integer (0 to 65535)
uint32	Unsigned integer (0 to 4.29E+9)
uint64	Unsigned integer (0 to 1.84E+19)
float16	Half precision signed float
float32	Single precision signed float
float64	Double precision signed float
complex64	Complex number: two 32-bit floats (real and imaginary components)
complex128	Complex number: two 64-bit floats (real and imaginary components)

Mảng nhiều chiều

(0-D: scalar)



Kiểm tra số chiều và kích thước mảng

`a.ndim`	Kiểm tra số chiều
`a.shape, np.shape(a)`	Kích thước mỗi chiều
`a.size, np.size()`	Tổng số items
`a.itemsize`	Kích thước mỗi phần tử (bytes)

1.2 Một số hàm tính toán, sắp xếp

Một số phép toán cơ bản:

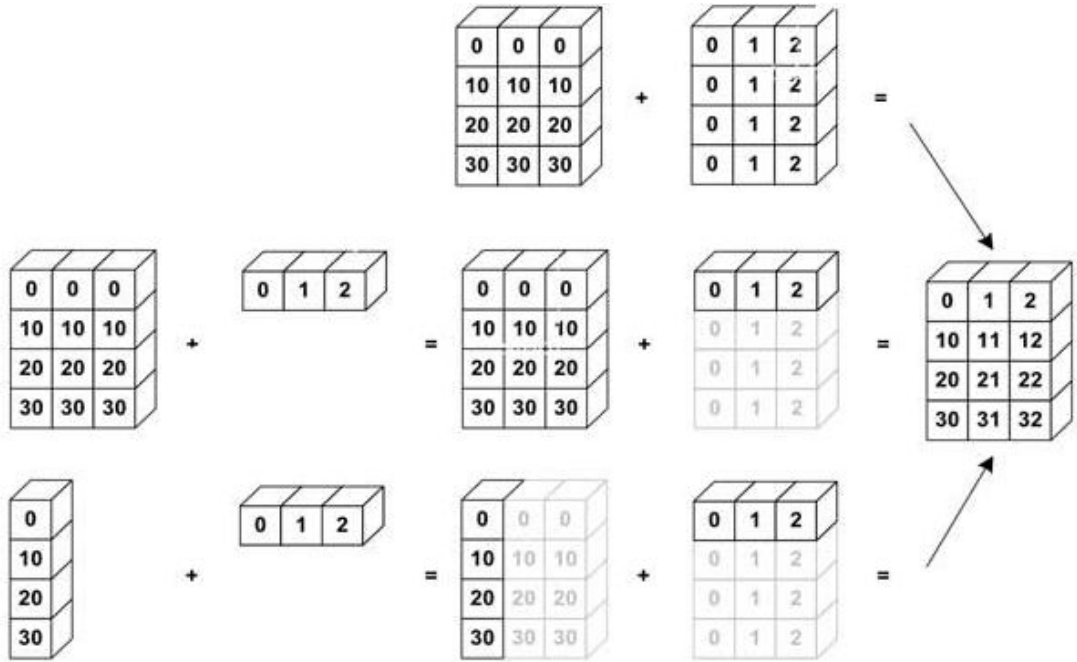
- Cộng/Trừ/Nhân/Chia (element-wise)
- Broadcasting
- Lượng giác: sin/cos/tan
- Làm tròn: round, around, floor, ceil, trunc (lấy phần nguyên, kể cả số âm)
- Tính tổng, tích và độ sai khác:
  - np.sum, np.nansum, np.cumsum (cumulative sum)
  - np.prod
  - np.diff

1.2 Một số hàm tính toán, sắp xếp

Một số phép toán cơ bản:

- Trị tuyệt đối: np.absolute, np.fabs
- Lũy thừa: np.exp, square, np.sqrt, np.cqrt,
- Logarithm: log, log2
- Phép nhân vector/ma trận: np.dot

❖ Broadcasting



Link: https://www.iodocs.com/difference-reshape-resize/

1.2 Một số hàm tính toán, sắp xếp

Các hàm thống kê cơ bản:

- Tổng/Max/Min/Range
  - np.max, np.amax, np.nanmax
  - np.min, np.amin, np.nanmin
  - np.ptp: range (maximum-minimum)
- Averages và Variance:
  - np.mean, np.nanmean, np.median, np.nanmedian
  - np.average (weighted average)
  - np.std, np.nanstd, np.var, np.nanvar
- Percentiles va Quantiles
  - np.percentile, np.nanpercentile
  - np.quantile, np.nanquantile
- Histogram: np.histogram

1.2 Một số hàm tính toán, sắp xếp

❖ Sorting

np.sort(a[, axis, kind, order])	Trả về một bản copy được sorted (kind = {'quicksort', 'mergesort', 'heapsort', 'stable'})
a.sort([axis, kind, order])	Phương thức để một mảng ndarray tự sort
np.argsort(a[, axis, kind, order])	Trả về indices theo thứ tự được sorted của a
np.lexsort(keys[, axis])	Sort các list cùng nhau và trả về indices (mặc định: sort theo list cuối cùng)

1.3 Một số hàm khởi tạo đặc biệt

Các hàm tạo nhanh một mảng thông thường:

- `numpy.zeros(shape, dtype=float)` và `numpy.zeros_like(a, dtype=None)`
- `numpy.ones(shape, dtype=float)` và `numpy.ones_like(a, dtype=None)`
- `numpy.empty(shape, dtype=float)` và `numpy.empty_like(a, dtype=None)`
- `numpy.full(shape, fill_value, dtype=None)`

- `numpy.arange([start, ]stop, [step, ]dtype=None)`
- `numpy.linspace(start, stop, num=50, endpoint=True, retstep=False, dtype=None, axis=0)`

1.3 Một số hàm khởi tạo đặc biệt

Một số hàm khởi tạo random:

```
from numpy import random
x = random.randint(0, 100)
y = random.rand([4,3])
z = random.normal(3.0, 1.5, [4,3])
t = random.uniform(1.0, 5.0, [4,3])
```

`random.random(size=None)`	Tạo một số hoặc mảng float ngẫu nhiên từ (0,1]
`random.rand(d0, d1, ..., dn)`	Tạo một số hoặc mảng float ngẫu nhiên (n-chiều)
`random.randint(low, high=None, /… size=None, dtype=int)` `random.random_integers(low, high)`	Các giá trị integer ngẫu nhiên theo phân bố đều: từ [low to high)
`random.normal(loc=0.0, scale=1.0, /… size=None)`	Các giá trị float ngẫu nhiên theo phân bố chuẩn
`random.uniform(low=1.0, high=1.0, /… size=None)`	Các giá trị float ngẫu nhiên theo phân bố đều

1.4 Tìm kiếm/Đếm

Searching

amax, nanmax amin, nanmin	Tìm giá trị lớn nhất, nhỏ nhất theo một chiều nào đó
argmax(a[, axis]) argmin(a[, axis])	Tìm vị trí lớn nhất, nhỏ nhất theo một chiều nào đó
nanargmax(a[, axis]) nanargmin(a[, axis])	Tìm vị trí lớn nhất, nhỏ nhất và bỏ qua giá trị NaN
where(condition, [x, y])	Chọn item từ x hay y tùy điều kiện
argwhere(a-and-condition)	tìm indices của các giá trị trong a thỏa điều kiện
nonzero(a)	tìm indices của các giá trị khác 0 trong a
searchsorted(a, v[, side, sorter])	tìm indices để insert v vào a mà vẫn đảm bảo thứ tự
extract(condition, a)	trả về các giá trị của a thỏa điều kiện nào đó
unique(a)	tìm các thành phần duy nhất trong mảng

Đếm

count_nonzero(a[, axis])	đếm các giá trị khác 0
bincount	đếm số lần xuất hiện của các thành phần trong từng bin (giống histogram)
unique(a, return_counts=True)	hàm unique có thể dùng để đếm số lần xuất hiện của các các giá trị trong mảng

Phần 2. Cơ bản về xử lý dữ liệu text

2.1. Cấu thành dữ liệu text
2.2. Các xử lý arrays trên chuỗi text
2.3. Một số phương thức cơ bản của chuỗi
2.4. Format chuỗi kết quả tính toán
2.5. Một số xử lý text với Numpy
2.6. Parse dữ liệu text sang dạng số

2.1 Cấu thành dữ liệu text

Một đối tượng dữ liệu text được đặt giữa 2 dấu nháy đơn '...' hoặc kép "..."

Một text có thể bao gồm tất cả các ký tự, trừ ký tự escape '\'
Khi in ra màn hình: '\n' sẽ tạo một dòng mới, '\t' sẽ thêm 1 Tab

Sau khi khai báo, giá trị của ký tự trong chuỗi không thay đổi được (immutable)

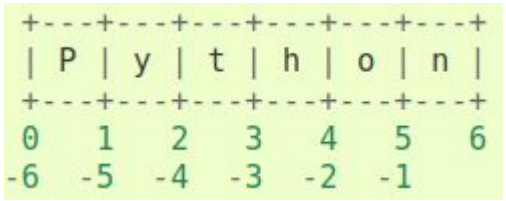
```
In [3]: print("Tôi tên là: \nTrần Văn A")
Tôi tên là:
Trần Văn A

In [4]: print("Tôi tên là: \n \'Trần Văn A\'")
Tôi tên là:
 'Trần Văn A'

In [5]: print("Tôi tên là: \n \"Trần Văn A\"")
Tôi tên là:
 "Trần Văn A"
```

2.2 Các xử lý arrays trên chuỗi text

- len()
- slicing
- looping



2.3 Một số phương thức cơ bản của chuỗi

Một số phương thức cơ bản của chuỗi:

- lower() / upper() / title()
- count()
- find()
- replace()
- split()
- Ghép chuỗi: phép cộng chuỗi, phép join()
- Kiểm tra chuỗi con: in

Tham khảo các phương thức trên chuỗi text:

https://www.w3schools.com/python/python_ref_string.asp

2.3 Một số phương thức cơ bản của chuỗi

- Hàm replace()
  cần phải gán kết quả vào một biến nào đó

```
string.replace(oldvalue, newvalue, count)
```

2.3 Một số phương thức cơ bản của chuỗi

- Hàm split()

  separator mặc định là khoảng trắng

```
string.split(separator, maxsplit)
```

2.3 Một số phương thức cơ bản của chuỗi

- Hàm join()

  dùng để nối một danh sách các string items thành chuỗi string theo một separator xác định trước

```
separator_string.join(list-of-strings)
```

2.4 Format chuỗi kết quả tính toán

```
1. Ký hiệu %: "%d %f %s"%(int_var, float_var, str_var)
2. Tiền tố f: f"{variable}"
3. Hàm format(): "{}".format(number)
```

2.5 Một số xử lý text với Numpy

Module **numpy.char**

Link: https://numpy.org/doc/stable/reference/routines.char.html#

`count(a, sub[, start, end])`	Đếm số lần xuất hiện của chuỗi con (sub) trong a
`find(a, sub[, start, end])` `index(a, sub[, start, end])`	Trả về vị trí đầu tiên (bên trái) của chuỗi con trong a (hàm index sẽ báo lỗi nếu không có chuỗi con)
`rfind(a, sub[, start, end])` `rindex(a, sub[, start, end])`	Trả về vị trí đầu tiên (bên phải) của chuỗi con trong a (hàm rindex sẽ báo lỗi nếu không có chuỗi con)
`startswith(a, prefix[, start, end])` `endswith(a, suffix[, start, end])`	Trả về một chuỗi boolean gồm True/False nếu các thành phần trong text bắt đầu hoặc kết thúc với prefix/suffix
`isalpha(a), isalnum(a), isdigit(a)` `isdecimal(a), isnumeric(a)`	Kiểm tra text có phải là số, chữ, chữ+số, số thập phân
`isspace(a), istitle(a)`	
`isupper(a), islower(a)`	

2.6 Parse dữ liệu text sang dạng số

Dùng hàm có sẵn trong Python

- int(), np.int()
- float(), np.float()
- np.fromstring(chuỗi-text, dtype, sep=''): parse một chuỗi text các số

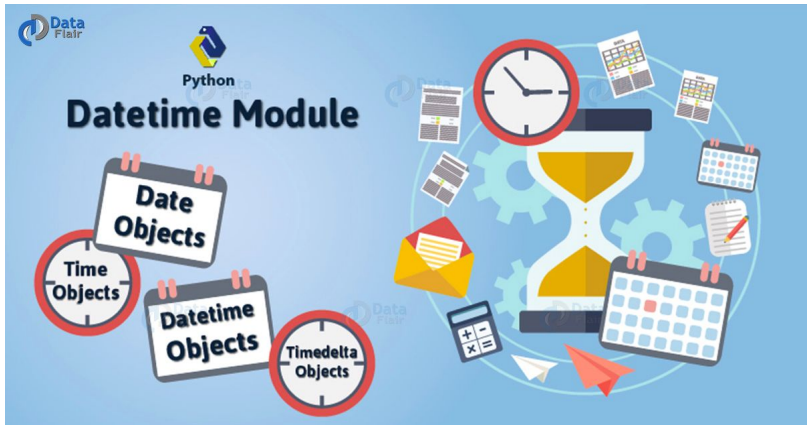
Ngược lại: dùng hàm str()

Phần 3. Cơ bản về dữ liệu thời gian

3.1. Các loại dữ liệu thời gian
3.2. Parse dữ liệu thời gian sang text và ngược lại
3.3. Các phép tính trên dữ liệu thời gian
3.4. Dùng numpy cho dữ liệu thời gian

3 Cơ bản về dữ liệu thời gian

- Dữ liệu thời gian thường được xử lý bằng thư viện datetime
- Để sử dụng, cần thực hiện import thư viện này




```
import datetime

x = datetime.datetime.now()
print(x)
```

```
2021-08-22 20:59:31.286850
```

```
import datetime as dt

x = dt.datetime.now()
print(x)
```

```
2021-08-22 21:00:40.259186
```

Tham khảo nâng cao: https://docs.python.org/3/library/datetime.html

`datetime` — Basic date and time types

Source code: Lib/datetime.py

The `datetime` module supplies classes for manipulating dates and times.

While date and time arithmetic is supported, the focus of the implementation is on efficient attribute extraction for output formatting and manipulation.

See also:

Module `calendar`
General calendar related functions.

Module `time`
Time access and conversions.

Package **dateutil**
Third-party library with expanded time zone and parsing support.

Aware and Naive Objects

Date and time objects may be categorized as “aware” or “naive” depending on whether or not they include timezone information.

With sufficient knowledge of applicable algorithmic and political time adjustments, such as time zone and daylight saving time information, an **aware** object can locate itself relative to other aware objects. An aware object represents a specific moment in time that is not open to interpretation. [1]

3.1 Các loại dữ liệu thời gian

`datetime.date`	Ngày trong năm theo Dương lịch, bao gồm year, month và day.
`datetime.time`	Giờ trong ngày, bao gồm hour, minute, second, microsecond, and tzinfo
`datetime.datetime`	Kết hợp ngày và giờ, bao gồm các thuộc tính của ngày và giờ
`datetime.timedelta`	là một khoảng thời gian diễn tả khoảng cách của date, time hoặc datetime
`datetime.tzinfo` `datetime.timezone`	Các lớp biểu diễn timezone

3.2 Parse dữ liệu thời gian sang text và ngược lại

- Chuyển đổi datetime sang text: strftime("format")
- Chuyển đổi text sang datetime: strptime("date_string", "format")
- Format code: https://docs.python.org/3/library/datetime.html#strftime-and-strptime-format-codes

Component	Code	Value
Year (as four-digit integer)	%Y	2020
Month (as zero-padded decimal)	%m	01
Date (as zero-padded decimal)	%d	31
Hour (as zero-padded decimal with 24-hour clock)	%H	14
Minute (as zero-padded decimal)	%M	45
Second (as zero-padded decimal)	%S	37

3.2 Parse dữ liệu thời gian sang text và ngược lại

- Chuyển đổi datetime sang text: strftime("format")

```
now = dt.datetime.now()
time_str = now.strftime("%Y-%m-%d %D %A %B %H:%M:%S")
print(time_str)
```

```
2021-08-22 08/22/21 Sunday August 22:01:30
```

3.2 Parse dữ liệu thời gian sang text và ngược lại

- Chuyển đổi text sang datetime: strptime(“date_string”, “format”)

```
a_dt = dt.datetime.strptime("15/08/2021 15:45", "%d/%m/%Y %H:%M")
print(a_dt)
```

```
2021-08-15 15:45:00
```

3.3 Các phép tính trên dữ liệu thời gian

Sử dụng timedelta để thực hiện các phép tính

```
timedelta(days=0, seconds=0, microseconds=0,
          milliseconds=0, minutes=0, hours=0, weeks=0)
```

```
from datetime import datetime as dt, timedelta
now = dt(2021, 9, 30, 1, 0, 0)
delta1 = timedelta(days=+2, hours=-3)
next1 = now + delta1
print(next1)

delta2 = timedelta(hours=45)
next2 = now + delta2
print(next2)
```

```
2021-10-01 22:00:00
2021-10-01 22:00:00
```

3.4 Dùng numpy cho dữ liệu thời gian

- np.datetime64
- np.timedelta64
- np.busday_offset
- np.is_busday
- np.busday_count

Tham khảo: https://numpy.org/doc/stable/reference/arrays.datetime.html

Bài tập tuần 05

Thực hiện các yêu cầu sau với dữ liệu '***vn_housing_dataset.csv***':

1. Lấy ra thông tin từ các cột Diện tích, Chiều dài, chiều rộng, Giá nhà
2. Kiểm tra các thông tin bất thường
   ⇒ Loại bỏ nếu thấy bất hợp lý
3. Chuẩn hóa thông tin (ví dụ: giá nhà với đơn vị "tỷ" thì đổi hết sang đơn vị "triệu", nếu có)
4. Lấy thêm thông tin từ cột Ngày tháng, format lại thành "Ngày_Tháng_Năm"
5. Lưu lại dữ liệu đã làm sạch, chuẩn hóa, loại bỏ bất thường… vào file csv khác.

THANK YOU!